Incentive-based Resource Assignment and Regulation for Collaborative Cloud Services in Community Networks

Amin M. Khan[a], Ümit C. Büyükşahin[b], Felix Freitag[a,]

[a] *Department of Computer Architecture, Universitat Politècnica de Catalunya BarcelonaTech, Barcelona, Spain*
[b] *Department of Computer Engineering, Middle East Technical University, Ankara, Turkey*

Abstract

Community networks are a successful example of a collective where communities operate ICT infrastructure and provide IP connectivity based on the principle of reciprocal resource sharing of network bandwidth. This sharing, however, has not extended to computing and storage resources, resulting in very few applications and services which are currently deployed within community networks. Cloud computing, as in today's Internet, has made it common to consume resources provided by public clouds providers, but such cloud infrastructures have not materialized within community networks. We analyse in this paper socio-technical characteristics of community networks in order to derive scenarios for community clouds. Based on an architecture for such a community cloud, we implement a prototype for the incentive-driven resource assignment component, deploy it in a testbed of community network nodes, and evaluate its behaviour experimentally. In simulations of large scale community cloud scenarios we study the behaviour of the incentive mechanism in different configurations. Our evaluation gives insight into how the developed mechanisms regulate the consumption of cloud resources taking into account the users' contributions, and how this regulation affects the system usage. Our results suggest a further integration of this regulation component into current cloud management platforms in order to open them up for the operation of an ecosystem of collaborative cloud services in community networks.

Keywords: cloud computing, community networks, incentive mechanisms, resource regulation

1. Introduction

Community networking is a shared communication infrastructure in which citizens build and own open communication networks. Most of these community networks are based on Wi-Fi technology such as ad-hoc networks or IEEE 802.11a/b/g/n access points in the first hop and long-distance point-to-point Wi-Fi links for the trunk network. Recently, a growing number of optical fibre links are also being deployed [1]. Despite the lack of reliable statistics, community networks seem to be rather successful. There are several large community networks in Europe, having from 500 to 20,000 nodes, such as Athens Wireless Metropolitan Network (AWMN), Freifunk.net, FunkFeuer.at, Guifi.net, Ninux.org, and many others worldwide. Figure 1 shows the wireless links and nodes of Guifi.net in the area around Barcelona.

The community cloud we present in this paper is the vision of a cloud deployment in community networks: A cloud hosted on community-owned computing and communication resources

providing services of local interest. The concept of community clouds has been introduced in its generic form before, e.g. [2, 3], as a cloud deployment model in which a cloud infrastructure is built and provisioned for an exclusive use by a specific community of consumers with shared concerns and interests, owned and managed by the community or by a third party or a combination of both.

Community networks successfully operate as IP networks, since the nodes' bandwidth is shared among all the members in a reciprocal manner. While there are also services offered from within the community networks, most members of community networks use the infrastructure solely to access the Internet, and they consume services in the Internet and not within the community network. If there are services inside the network, they usually run on machines exclusively used by a single member (normally the owner of the machine). We emphasize that the sharing of storage and computational resources, which is now common practice in today's Internet through cloud computing, hardly exists in community networks.

Community networks are an ecosystem which is able to regulate and maintain itself, some of the community networks are there for even more than a decade. Participants of the community network not only contribute infrastructure to the network, but also their knowledge, time and effort for successful operation of the network. We anticipate that cloud infrastructures for community networks will need additional incentive mechanisms in order to achieve sustainability. In this paper we study an incentive mechanism for clouds in community networks, keeping in view the key characteristics of community networks and the scenarios we foresee for community clouds. This incentive mechanism is inspired by Parecon economic model [4, 5], and based on the idea of effort of each participant, which we define as its contribution relative to its capacity. Our approach is to do the evaluation with a prototype and simulation experiments, which will allow us to derive additional conclusions regarding its feasibility for implementation and deployment on a wider scale. The main contributions of this paper are the following:

1. We identify a community cloud scenario, envisioned as a federation of local clouds, which is derived from a socio-technical analysis of community networks.
2. We implement a proposed incentive mechanism in a regulation component and deploy it in real nodes of a community network.
3. We evaluate experimentally with the deployed prototype the behaviour of the incentive-driven resource assignment in the community cloud scenario.
4. We evaluate the behaviour of incentive mechanism in simulations of large scale community cloud scenarios with different configurations.

We elaborate our contributions in the following way. In section 2, we analyse community networks and bring about the community cloud scenario. In section 3, we discuss a cloud architecture applicable to the topology of community network deployments, taking into account socio-economic context of the community networks necessary for encouraging collaborative resource sharing. In section 4, we introduce the prototype implementation for the resource assignment component of the community cloud architecture, and we evaluate in experiments the resource assignment behaviour of the prototype by deploying it in a testbed of real community network nodes. In section 5, we evaluate our incentive mechanism with simulation experiments in a community cloud scenario. In section 6, we present related work, and in section 7, we conclude and indicate future work.

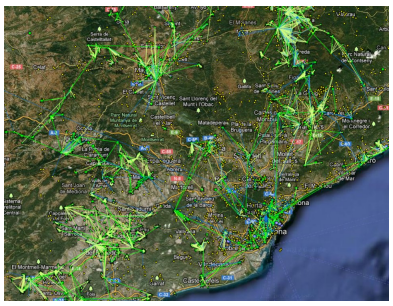

Figure 1: Guifi.net nodes and links in Barcelona

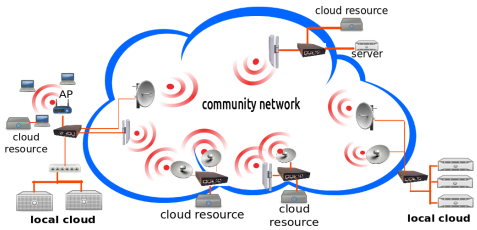

Figure 2: Nodes in a community network

2. Clouds in Community Networks

Since our community cloud is targeted to be used in real community networks, it is a must that our architecture, design, implementation and deployment fits into these conditions and scenarios. We focus our analysis on the Guifi.net community network, which is considered the largest community network worldwide, and it is where we have also deployed our prototype.

2.1. Community Networks

2.1.1. Nodes and topological aspects of community networks

A community network is managed and owned by the community, where nodes are managed independently by their owners. The computer machines or nodes in a community network vary widely in their capacity, function and capability, as illustrated in Figure 2. Some hardware is used as super nodes (SNs) that have multiple wireless links and connect with other SNs to form the backbone of the community network, and are usually intended to be stable with permanent connectivity. Most SNs are installed in the community network participant's premises. A few SNs, however are placed strategically in a third party location, e.g. telecommunication installations of municipalities, to improve the community network's backbone. Other nodes in the community network act as ordinary nodes (ON) and are only connected to the access point of a SN. Topological analysis of the Guifi.net community network [6] indicates that from approximately 17,000 analysed nodes of Guifi.net, 7% are SNs while the others are ONs.

From the node types shown in Figure 2, it can be seen that principally the hardware for computation and storage is already available in community networks, consisting of some servers attached to the networking nodes. No cloud services, however, are yet deployed in community networks to use this hardware as a cloud, leaving the community network services significantly behind the current standard of the Internet. Our vision is that some community wireless routers will have cloud resources attached, building the infrastructure for a community cloud formed by several cloud resources attached to the nodes. We note that ONs could principally also contribute cloud resources.

Figure 3 shows the outdoor view of a community network SN. The equipment, mainly antennas and radios, is used for building wireless links between other SNs. Figure 4 shows an example of the indoor hardware of a SN. The router used is a Microtik RB750, while a Jetway JBC362F36W with Intel Atom N2600 CPU, 2GB RAM and 64GB USB has been added to become a cloud resource

Figure 3: SN with outdoor hardware for wireless links

Figure 4: Indoor hardware of a community network node with router, server and cloud resource

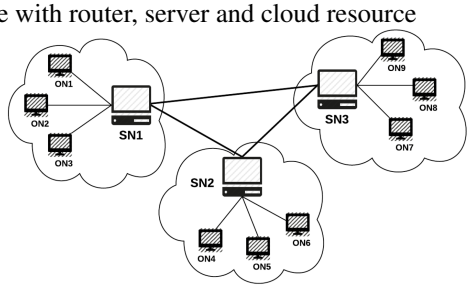

Figure 5: Nodes in federated community cloud

for the community cloud. A laptop is used as an additional server, while a UPS keeps the node running in the case of power failure.

2.1.2. Social aspects of community networks

Personal and social relationships play an important role in the community network deployment. The deployment of new nodes requires the collaboration among people. If a new node is deployed, the owners of the neighbouring nodes need to connect with it, thus there has to be an interaction among the people. Two types of social networks can be observed from Guifi.net's mailing list [7]. One is at the global level of the whole Guifi.net network. In this list, technical issues are discussed. People from any part of Guifi.net community participate, and even external people who are interested can take part. The second type is the local social network, between node owners within a zone and between neighbouring zones. They use local mailing lists as well as hold weekly meetings.

Guifi.net is organized into zones. A zone can be a village, a small city, a region, or a district of a larger city. The organization of the group within a zone is of many types. Mostly the interests, available time and education of the people drive what happens in the zone. We note that while the allocation of IP addresses and layer 3 networking is agreed among all Guifi.net zones, as it is needed to make the IP network work, the detailed technical support is rather given within the local community of the zone. Therefore, we identify a zone to have the highest social strength within the community network.

2.1.3. Members of community networks

Participants of community networks are principally consumers and producers of the network. Most of them as producers contribute infrastructure and time to the networks, while as consumers they use the available services the network offers. The community network, however, is not maintained solely based on the contribution of infrastructure. Some users must also contribute with their time and knowledge. Time is needed, for instance, for maintenance tasks, which might require

technical knowledge or not. Technical knowledge is required because the network is an IP network, which needs to be managed and configured.

2.1.4. Resource sharing in community networks

Community networks are a successful case of resource sharing among a collective. The resources shared are networking hardware but also community network participants' time that they donate, to different extent, for maintaining the network. While the community network infrastructure is the sum of the individual contributions of wireless equipment, the network operation is achieved by the contribution of time and knowledge of the participants. This is because even under the decentralized management of the equipment, the owner of the device ultimately has the full access and control of that network device.

Reciprocal resource sharing is, in fact, part of the membership rules or peering agreements of many community networks. The Wireless Commons License (WCL) [8] of many community networks states that the network participants that extend the network, e.g. contribute new nodes, will extend the network in the same WCL terms and conditions, allowing traffic of other members to transit on their own network segments. Therefore, resource sharing in community networks from the equipment perspective refers in practice to the sharing of the nodes' bandwidth. This sharing, done in a reciprocal manner, enables the traffic from other nodes to be routed over the nodes of different node owners and allows community networks to successfully operate as IP networks. We observe that in most community networks the focus at the moment is on the bandwidth sharing alone. There is not much awareness about sharing other computing resources, such as storage or CPU time, inside of community networks.

2.1.5. Ownership of nodes in community networks

Community networks grow organically. Typically a new member that wants to connect to the community network contributes with the hardware required to connect to other nodes. A node of a community network therefore belongs to the member who is its sole owner. Such a node is normally located in the member's premises.

Although less typical, a few nodes in Guifi.net have also been successfully crowd-funded if such a node was needed by several people. Crowd-funding of a node happened when for a group of people an infrastructure improvement was necessary. For example, an isolated zone of Guifi.net established a super node to connect to other zones. In such a case, the node has been purchased with the contributions of many people. The location of such a node follows strategic considerations, trying to optimize the positive effects on the performance that are achieved with the addition of the new infrastructure. We can see that both the options, individual ownership and crowd-funding of resources, occur in practice and could be considered for community clouds.

2.1.6. Services in community networks

Services and applications offered in community networks usually run on the machines that the member connects to the network and these machines are used exclusively by that member. The usage of the community network's services among its members, beyond that of access to the Internet, is however not very strong.

2.2. Community Cloud Scenarios

Based on the socio-technical characteristics of community networks analysed above, we start sketching our vision of community clouds. The scenario of *local community cloud* is derived from the topology of the community network, given by the fact that the community network generally has two different types of nodes, SNs and ONs, and the observed characteristics of the strength of social network within zones [6]. In such a local community cloud, a SN is responsible for the management of a set of attached nodes contributing cloud resources. From the perspective of the attached nodes, this SN acts as a centralized unit to manage the cloud services.

Multiple SNs from different zones in a community network can connect and form *federated community cloud* [9]. SNs connect physically with other SNs through wireless links and logically in an overlay network to other SNs that manage local clouds. SNs coordinate among themselves for provisioning infrastructure service so the requests originating from one SN's zone can be satisfied by the resources allocated from another SN's zone. Figure 5 shows an example of a federated community cloud formed by SNs from three zones.

2.3. Social and Economic Mechanisms for Supporting Collaboration

The purpose of economic mechanisms and social and psychological incentives is to let the community cloud transition from inception through early adoption to finally ubiquitous usage [10]. Such mechanisms must take into account the costs and benefits involved in participating in community cloud. For instance, the initial costs for setting up nodes in the community cloud involves hardware costs including the price of the computing and networking equipment, and installation costs including the manual labour needed. Besides these costs at the individual level, there are also the transaction costs or management overheads to direct the group coordination and collaborative production efforts necessary for the operation of community cloud. The individuals in community cloud act as private enterprises where they offer services to generate revenue. The revenue for the community cloud users include tangible benefits like the services and applications that they will be able to consume, and intangible benefits like the sense of belonging to the community and personal satisfaction because of their contributions. The services can range from infrastructure to platform to software services meeting a spectrum of different needs of the users.

Different policies addressing relevant issues of the technical, social, economic and legal aspects of the community cloud are designed to encourage collaboration, for example commons license and peering agreements can be implemented that extend the idea of reciprocal sharing from Wireless Commons License [8] and Pico Peering Agreement [11] in community networks. The social context of community networks provides opportunity to harness social capital and the different roles of social relationships. Similarly, lowering transaction costs and entry barriers, facilitating participation of developers, exploring different service models to provide value addition and differentiation, and taking advantage of locality and overlay topology of the network can prove useful. Such mechanisms help adapt the ecosystem of community cloud infrastructure and services to the aspirations of the community network members.

3. Architecture and Design

The option for enabling a community cloud in a community network on which we focus here is to deploy on SNs a cloud management system tailored to community networks. Available popular

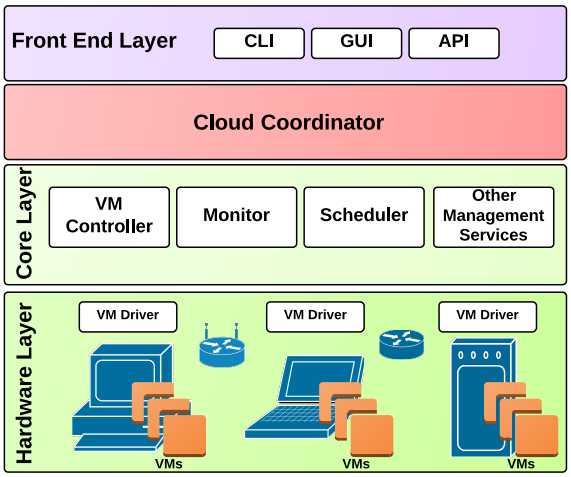

Figure 6: Architecture of cloud management system

cloud management platforms, notably OpenStack [12] and OpenNebula [9] among others, can principally be applied to provide the basic management of local clouds. Such cloud management systems can be tailored for community networks by extending the existing functionality to address the particular conditions of community networks. For example, incentive mechanisms inspired by the social nature of community networks can be built into resource regulation component to encourage users to contribute resources [13, 14].

3.1. Community Cloud Management System

The architecture for the cloud management platform that we propose for community networks consists of multiple layers [15], with different components at each layer, as shown in Figure 6.

3.1.1. Hardware Layer

This consists of the physical infrastructure that is needed to run a cloud system. The hardware in the community networks mostly consists of ONs and SNs and the communication infrastructure, along with any attached computation, storage and other resources.

3.1.2. Core Layer

The core layer consists of components that are responsible for creation, allocation, scheduling, monitoring and management of VMs on the nodes. The functionality of the core layer is already provided by tools like OpenStack and others. Community cloud manager can, therefore, make use of these existing tools and extend their functionality to suit the needs of the community network.

3.1.3. Cloud Coordinator

The cloud coordinator is responsible for the federation of the cloud resources which are independently managed by different SNs. The cloud coordinator components in different SNs connect among themselves in a decentralised manner to exchange relevant information about managing the available resources. Normally applications running at a local community cloud can only consume resources from the ONs directly managed by that particular SN. With the cloud coordinator, the

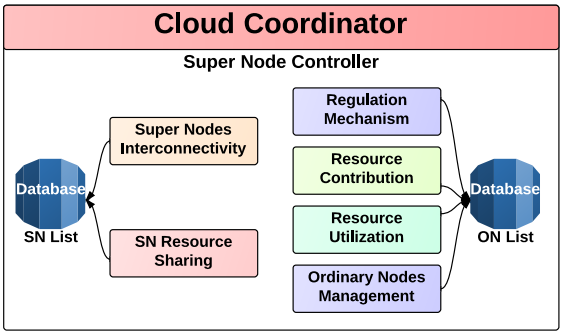

Figure 7: Components of cloud coordinator

infrastructure service can provide a unified view of the resources contributed by multiple local community clouds. Figure 7 shows the implemented components of the cloud coordinator in the prototype as described below.

- ON Management: ONs can register with SN to request and to contribute resources.
- Regulation Mechanism: When pooling resources from multiple zones, the cloud coordinator applies a regulation mechanism that takes into account resource utilization and contribution by different nodes to perform resource allocation.
- SN Interconnectivity: The design of a community cloud manager follows a decentralized approach, so cloud coordinator relies on gossip-based discovery mechanisms to manage overlay network of the SNs in community cloud. The updated list of adjacent SNs is saved in SN-List database.
- SN Resource Sharing: When requests from ONs cannot be met from locally available resources, SN can request resources from other SNs in the system.

3.1.4. Frontend Layer

The frontend layer provides the interface to interact with the infrastructure service of the community cloud. This includes modules like command line interface (CLI), graphical user interface (GUI), application programming interface (API), and any other tools that assist with developing cloud application using the infrastructure service.

3.2. Interaction between Super and Ordinary Nodes

The overlay network that results from the hierarchical architecture of the community cloud is formed by SNs. The difference between SNs and ONs, from the point of view of cloud management, is that SNs support greater functionality for handling VMs. A SN has full installation of the cloud management software and so enables the user to manage VMs executing on ONs. In most cases, a SN will be a comparatively stable node, most likely connected to a hub of the wireless mesh network. Each SN is responsible for a set of ONs and and manages their metadata in its ON-List

database. SNs publish their status and details of local resources to other SNs, e.g. by gossiping, and each SN stores this information in SN-List database. ONs, on the other hand, only act as hosts for executing VMs. Most components of cloud management software are not installed on ONs, so the VMs cannot be controlled conveniently from the ONs themselves. Each ON registers to a parent SN by providing it with the list of the available resources, and the parent SN is responsible for management of VMs. ONs periodically send a heartbeat message to their parent SN to inform it about their current status.

3.3. Incentive Mechanisms in Community Cloud

Participants in a community network are mainly volunteers that act independently and are not obliged to contribute. To ensure sustainability and growth of the community cloud, incentive mechanisms are needed that encourage members to contribute with their hardware, effort and time [16]. When designing such mechanisms, the heterogeneity of the nodes and communication links has to be considered since each member brings in a widely varying set of resources and physical capacity to the system. Most peer-to-peer (P2P) systems implement incentive mechanisms based on contribution where nodes are rewarded according to resources they donate to the system [17]. We suggest an effort-based incentive mechanism for community cloud where effort is defined as contribution relative to the capacity of a node [13, 14]. This mechanism is inspired by the Parecon economic model [4, 5] which focuses on social welfare by considering inequality among nodes. Nodes with different capacity cannot have same contribution to the system but in this mechanism they get same reward if they share as much as possible of their capacity, as we explain in the following. We use a system of credits, acting as virtual currency, to facilitate transactions between providers and consumers. When resources are consumed, providers earn credits which they can later use to request resources from the system.

3.3.1. Formulations

We first discuss here the criteria that a SN uses to evaluate requests from ONs. When a node asks for a resource from a SN, which in this case means to commit an instance of VM for a given duration, the SN first checks whether the ON's credit is sufficient to cover the cost of the transaction. The cost is proportional to the number of resources requested R_i and the duration T_i for how long they are required.

$$transaction\_cost = \gamma R_i \times \rho T_i \tag{1}$$

where γ and ρ are nonzero coefficients for the amount and duration of resources shared respectively. The coefficients γ and ρ provide a way to tweak the value generated by the transactions, and are useful to control the behaviour when implementing the prototype system. The cost from all the past transactions node i participates in, either as a provider or a consumer, determines the level of its credit.

If the requesting node does not have enough credit, the request is rejected. Otherwise, the SN searches for nodes that have resources available. It selects as many nodes as possible from its local zone as providers. If the demand cannot be met locally, the SN forwards the request to super nodes in the federated community cloud.

Now we consider how the SN manages the credits of the nodes that take part in the transaction. For each node which contributed its resources to fulfil the request, the SN calculates the transaction cost as shown above and adds it to that node's credits. The cost is deducted from the credits of the node that consumed the resources. After the transaction is completed, the effort for each node involved in the transaction is recalculated as:

$$E_i = \begin{cases} \frac{credit_i}{\epsilon C_i} & if \quad \frac{credit_i}{\epsilon C_i} < 1 \\ 1 & otherwise \end{cases} \tag{2}$$

where ϵ is nonzero coefficient for the capacity of the node, and ϵ acts as a normalizing factor taking into account overall capacity in the system. A selfish node not contributing enough has effort $E_i < 1$, and will be at a disadvantage when requesting resources, as in equation 4 below.

The effort of a node expresses its relative contribution to the system, since the mechanism considers the capacity C_i of a node as well. This means that a node with low capacity puts in more effort than a node with high capacity even if both of them donate same amount of resources to the system. For total N nodes in the system, the total amount of available resources Ω is the sum of the resources ω_i contributed by each node $i \in N$. The maximum resources node i can consume, ΔR_i, depends upon its effort E_i. A node actively contributing to the system has $E_i = 1$ and so can access all the available resources $(\Omega - \omega_i)$, but a selfish node with $E_i < 1$ gets penalized with limited access to the resources.

$$\Omega = \sum_{i \in N} \omega_i \tag{3}$$

$$\Delta R_i = E_i \times (\Omega - \omega_i) \tag{4}$$

3.3.2. Algorithm for Requests Processing

Algorithm 1 explains how a SN handles request from a node in its zone. When SN receives a request, it first calculates that node's allowance ΔR_i to confirm whether it has enough credit to fulfil the request. If not, the request is rejected, otherwise the algorithm calls `decision` function which searches for available resources (lines 1–5). The `decision` function first checks if enough resources are available in the local zone (line 8), and selects the nodes that will provide the resources from its local zone (line 9). If SN cannot satisfy request from its local nodes, it forwards request to one of its neighbouring SNs (lines 16–18). After the provider nodes commit resources, SN calculates cost of the transaction and updates the nodes' credits, deducting credits from the requester and increasing credits of the providers (lines 10–14). The sequence of steps is depicted in Figure 8.

4. Prototype Implementation

We have implemented a prototype of the incentive-based regulation mechanism explained above [13], in Python using CouchDB [18] database at the back-end. We chose Python because the current host operating system installed on ONs is OpenWRT [19], which supports Python, but does not support many other languages such as Java. We selected CouchDB because among its advantages, it is lock-free, schema-less and provides a REST interface, and is also part of the other components of the SN's cloud management software being developed. In the SNs, Debian operating system is installed.

Algorithm 1 Handling requests from ONs

Require: receive query from node i with the requested amount R_i and the time T_i

```
calculate(ΔR_i)
if R_i <= ΔR_i then
    call Decision(i, R_i, T_i)
else
    send("rejected", i)
end if
function DECISION(i, R_i, T_i)
if R_i <= Ω then
    ProvidersList[n] ← provider(ON_List, R_i)
    for each j in ProviderList[n] do
        CostOfTransaction_{j→i} ← R^r_j * T^t_j
        update_credits(CostOfTransaction_{j→i})
        update_database(ON_List)
    end for
else
    SN ← provider(SN_List,R_i, reserved_ratio)
    forward(SN,i, R_i, T_i )
end if
```

ONs use the remote procedure call (RPC) mechanism to connect to the SN. First of all, an ON assigns itself to a parent SN with a register message which includes metadata of that ON such as IP address, total capacity and number of VMs shared. This registration information is stored in the ON-List database of the parent SN by creating an entry for the corresponding ON. After that, the ON is ready to send requests to its parent SN, which are processed using Algorithm 1 as shown in Figure 8. When an ON requests its parent SN for any VMs, it specifies the duration for how long it needs to use the VMs. This request is evaluated by performing incentive and decision mechanisms as explained in section 3. If a request cannot be met locally, the corresponding parent SN checks its SN-List database to find another zone with available resources. The interactions between SNs are also made through RPC mechanism. In the SN controller software, there is a separate process which regularly checks the database for any updates. If the duration of a consumer ON's resource request has expired, it frees the VMs and makes them available for the provider ON, and updates the metadata entries of the corresponding ONs in the ON-List database. The current implementation keeps track of the number of VMs contributed and consumed by each ON. The system copes with ONs connecting and disconnecting from the SN at any time since ONs periodically send heartbeat messages to the SN. The design allows us to include values of metrics like CPU, memory and bandwidth usage in the future for fine-grained decisions about resources assignment.

4.1. Evaluation

We deploy the prototype of the regulation component of the cloud coordinator from community cloud management system in the Community-Lab testbed, which is developed by the CONFINE European project [20]. The cloud coordinator components are installed on nodes of the Community-Lab testbed, which consist of devices of the model Jetway JBC372F36W, as introduced in Figure 4. Depending on the experiment, one or two nodes operate as SNs, while each ON hosts between one and four VM instances. The objectives of the experiments are twofold:

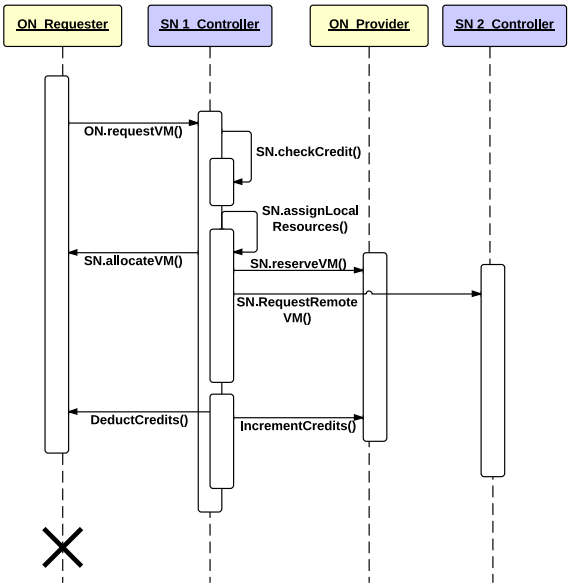

Figure 8: Details of the VM request operation by ON

1. Experiment 1: Assess the prototype operation regarding the incentive-based resource assignment algorithm in a local community cloud scenario.
2. Experiment 2: Study the coordination between SNs from different zones in the federated community cloud scenario with heterogeneous resource distribution.

4.2. Experiment 1: Resource Assignment in Local Community Cloud Scenario

In order to study the performance of the prototype in a real deployment of a local community cloud, we install our software components in four ONs and one SN in Community-Lab testbed, which are connected to the Guifi.net community network. Each node behaves as an ON but with different configuration, in order to have a heterogeneous set of cloud resources. The four nodes include f101 sharing 1 out of total 2 VMs, f102 sharing 3 out of total 3 VMs, f103 sharing 1 out of total 3 VMs, and f104 sharing 1 out of total 1 VM. Each ON sends request for VM instances to the SN at regular intervals. VMs are requested for 20 seconds interval at a time. Each ON requests as many VMs as its total capacity, for example node f101 always requests 2 VMs. If the request is accepted by the SN, the ON obtains the VMs for the next 20 seconds. If the request is rejected, the ON waits for 5 seconds before making any further requests. The experiment is run with this setup for around 5 minutes. We analyse the different aspects of the system behaviour in the following.

4.2.1. Resource Utilization

Figure 9 shows the level of resource utilization in the system in terms of the number of reserved VMs versus the total number of VMs. It can be seen that resource utilization varies widely and 100% utilization, meaning all the VMs being occupied, occurs only for short intervals. This is because as nodes obtain VMs, they spend their credit and can no longer request more VMs. At approximately second 80, the utilization gets very low. Nodes then need to earn credits by providing

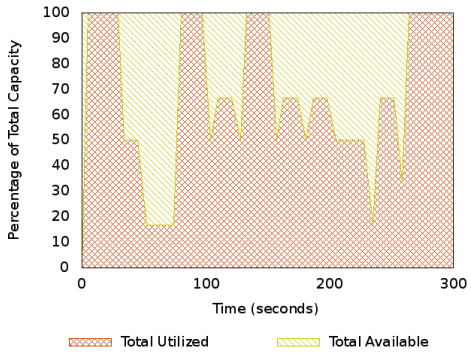

Figure 9: Overall resource utilization of the four ONs

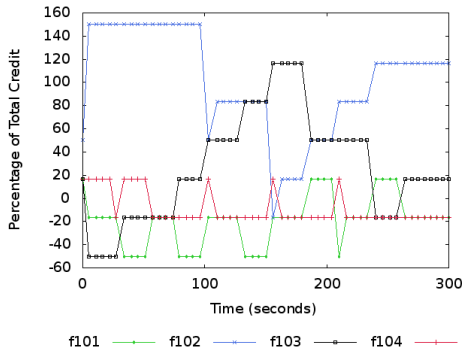

Figure 10: Distribution of credit among the four ONs

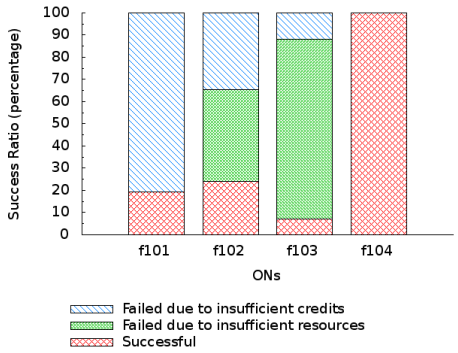

Figure 11: Ratio of fulfilled and rejected requests

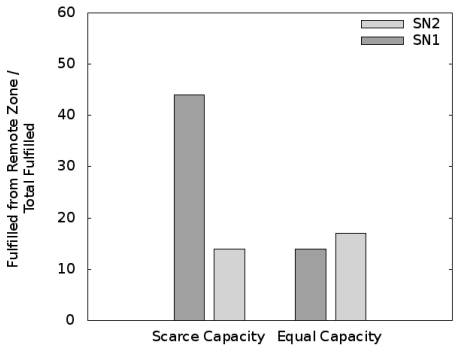

Figure 12: Resource assigned over different SN zones

VMs to others before they can request VMs again. So even though VMs are available, they cannot be utilized due to the lack of credit in the system.

4.2.2. Credit Distribution

Figure 10 shows the credit distribution among the four ONs during the 5 minutes of the experiment. A node's credit is affected by how many VMs it shares and how much credit it spends to obtain VMs. When a node shares most of its capacity, like ON f102 providing all its 3 VMs, it earns more credit and so maintains a high credit level during the experiment. On the other hand, when a node continuously consumes VMs like ON f101 and f104, it keeps on spending its credit which does not go beyond a certain level. Of particular interest is the behaviour of ON f103, which earns credit in the start and gets a spike in credit level halfway through the experiment, but then quickly spends it as it requests VMs from others. Note that an ON's credit can be negative or higher than 100% of the total credit because in the current implementation SN can allow requests from ONs with zero or negative credit.

4.2.3. Success Ratio

Figure 11 shows the ratio of the fulfilled requests for each node, which is affected by the level of credit of the node and the amount of resources available in the system. ON f104 has the most success since it requests only one VM at a time while ON f103 has the least success since it requests

Table 1: Two cases with different resource distribution between zones

		Case 1: Scarce Capacity		Case 2: Equal Capacity	
SNs	ONs	Total VMs	Shared VMs	Total VMs	Shared VMs
SN1	ON1	3	1	3	2
	ON2	3	1	3	3
	ON3	3	1	3	2
	ON4	1	1	1	1
SN2	ON1	3	2	3	2
	ON2	3	3	3	3
	ON3	3	2	3	2
	ON4	1	1	1	1

3 VMs, which is half of the total shared VMs in the system. ON f101, on the other hand, gets its requests rejected because of the lack of credit. Therefore, this node has to wait to gain the needed credits.

4.3. Experiment 2: Resource Assignment in Federated Community Cloud Scenario

In this experiment, we set up two local clouds, each with one SN and four ONs to study the federated community cloud scenario, as illustrated in Figure 5. Table 1 shows the two cases with different number of VMs available in the two zones. In the case of scarce capacity (case 1), the nodes in the SN1 zone share very few VMs compared to nodes in SN2 zone. In the case of equal capacity (case 2), the nodes in both the zones share the same number of VMs.

Figure 12 shows the proportion of the requests fulfilled by VMs provided by the other zone. With scarce capacity in SN1 zone, around 50% of the requests are fulfilled by VMs provided by SN2 zone. SN2 with sufficient capacity is able to meet most of the requests from VMs within the same zone, forwarding less than 15% requests to the other zone. In the second case, when both zones have the same available capacity, most of the requests get processed within the same zone for both the SNs. This shows that a federated community cloud scenario extends the resources assigned to zones with limited capacity.

4.4. Discussion

From the results of these experiments, we observed that:

1. The prototype of the regulation service deployed in real community network nodes performed the required operations. Its components worked correctly both in the ON's host operating system (OpenWRT) and the SN's operating system (Debian). We could not observe the limitations of our implementation within scales that are realistic for community networks. We note however that as a continuation of this work a more extensive deployment of several federated community clouds with real users and real usage should ultimately be undertaken.
2. The algorithm used for the regulated resource allocation controlled the VM assignments, taking into account the user's contribution and usage. More complex situations, however, should be created in further studies to provide additional insight into how the system behaves.

Table 2: Configuration for each node in a zone with shared and total instances

Node Behaviour	Shared	Small capacity	Medium capacity	Large capacity
Selfish	33%	ON1 (1/3)	ON2 (2/6)	ON3 (3/9)
Normal	66%	ON4 (2/3)	ON5 (4/6)	ON6 (6/9)
Altruistic	100%	ON7 (3/3)	ON8 (6/6)	ON9 (9/9)

3. Our experiments were carried out on limited number of nodes and for limited time. If our prototype was deployed on additional nodes that are geographically widely spread and run for extended periods, the VM assignment decisions might need to take into account information from network awareness, to select the appropriate cloud resource providers.

5. Evaluation with Simulation Experiments

In this section, we extend our study from the prototype implementation with simulation experiments that cover resource regulation on a larger scale across multiple SN zones covering both local and federated community cloud scenarios.

5.1. Experiment Setup

We simulate a community network comprising of 1,000 nodes which is divided into 100 zones and each zone has one SN and nine ONs. The zones are distributed in a small world topology where each zone is neighbour to 10 other zones. This approximation holds well for real world community networks as, for example, topology analysis of Guifi.net [6] shows that the ratio of super node to ordinary nodes is approximately 1 to 10. Each ordinary node in the simulation can host a number of VM instances that allows users' applications to run in isolation. Nodes in the zone have two main attributes, one is capacity which is the number of available VM instances, and other is sharing behaviour which is how many instances are shared with other nodes. Table 2 shows the different configurations for each of the nine ONs in each zone. Nodes with low, medium and high capacity host 3, 6 and 9 VM instances respectively and they exhibit selfish, normal or altruistic behaviour sharing one-third, two-thirds or all of their VM instances. For example, node ON2 has medium capacity with 6 instances and exhibits selfish behaviour reserving 4 instances for itself and contributing only 2 to the system.

When the experiment runs, nodes make requests for resources proportional to their capacity asking for two-thirds of their capacity. For instance nodes with capacity of 3, 6 and 9 VM instances request 2, 4 and 6 instances respectively. Nodes request instances for fixed duration and after transaction is complete wait briefly before making further requests.

5.2. Experimental Results

We evaluate the impact of the effort-based incentive mechanisms in the system in simulation experiments and discuss the results below. We study the success ratio, i.e. number of requests fulfilled versus total requests, and the overall resource utilization in the system.

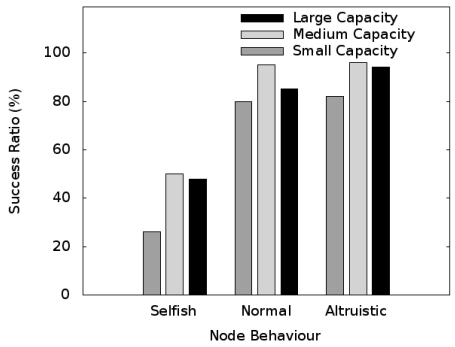

Figure 13: Success ratio of nodes for different configuration

5.2.1. Ratio of Successful Requests

Figure 13 shows the success ratio for requests made by different nodes analysed with the effort-based incentive mechanism. We first notice that this mechanism rewards the nodes which tend to share most of their capacity with the system, while penalizing the nodes which are less willing to share their resources. For example, even though selfish nodes with large capacity have a 48% success ratio, altruistic nodes with small capacity have twice as much at 82%.

Moreover, the values for success ratio decrease as the capacity of the nodes increases. This is explained by the fact that nodes with greater capacity request more instances and so they have a higher chance of getting rejected either because there are not many resources available in the system or because the requesting nodes do not have sufficient credit. However, when comparing the success ratio for nodes as their capacity increases, we observe that there is not a great variation. For instance, for the selfish sharing behaviour success ratio ranges from 26% to 48%. This is explained by the fact that the effort-based approach does take heterogeneity of nodes into account and rewards nodes with different capacity. As a result, our evaluations indicate that effort-based incentives ensure fairness in the system since the nodes with the same sharing behaviour are treated equally irrespective of their capacity.

5.2.2. Breakdown of Request Responses

Figure 14 shows the overall breakdown of successful and rejected requests across all the zones, where there are many more successful requests than rejected ones. Moreover, very few requests are rejected because of a lack of resources. This indicates that the effort-based incentive mechanism improves efficiency as more resources are utilized. In addition to this observation, the majority of the requests are fulfilled using resources from the local zone with very few requests forwarded to other zones.

5.2.3. Resource Utilization

Figure 15 shows the proportion of resources utilized in the system along the execution of a 24 minutes experiment. In the start all the nodes have enough credit and the resource utilization is high. Then it drops to below 50% at around the 8^{th} minute, and keeps fluctuating for a while. Afterwards, since most of the nodes have completed their transactions and consumed their credits, the utilization decreases significantly.

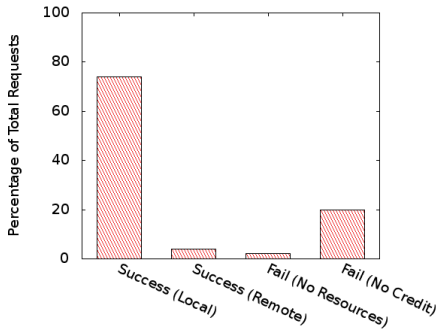

Figure 14: Breakdown of outcome of requests

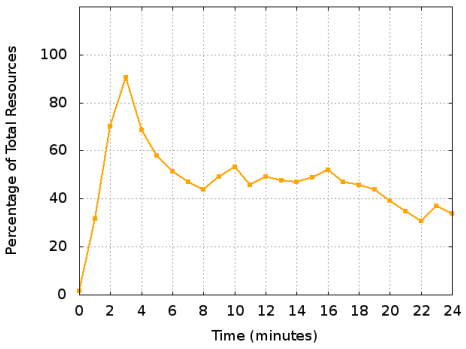

Figure 15: Resource utilization

5.3. Discussion

We have investigated incentive mechanisms for community clouds based on reciprocal resource sharing, and the results highlight their impact on the efficiency of the system and on regulating the resource assignments. The understanding gained from the different experimental results helps in the design of the policies that such incentive mechanism could follow in a future platform of real community cloud system.

Our results, however, have revealed new issues that are to be addressed in the next steps towards a real cloud system. In this line, the behaviour of the incentive mechanism for extended periods of time has not been investigated yet, so, further experiments should study how the mechanism can be used for long durations.

For the permanent operation of the cloud system with the incentive mechanism, the mechanism needs to be able to adapt to the system state in runtime. The mechanism will need to be able to take into account the evolution of the system with regards to users, resources, and different kind of behaviours. Therefore, parameters of the incentive mechanism will need to be defined as functions of the system state in order to account and decide correctly on the current situation. In order to further develop this runtime adaptability, a two-fold approach, which on one hand extends the simulations with refined system models and on the other hand evaluates the performance of deployed prototype components, is suggested to assure the realization of an operative adaptive system.

In order to able to obtain performance results from real users and services, a prototype of the incentive mechanism integrated in a cloud management platform is required. An operative modular system is needed that allows an easy modification of its components according to the simulation results. The transfer of the simulation results to the deployed system should be required, in order to assure that the simulated system model reflects the real system, and that the obtained findings can actually be brought into the real system in a feasible way.

Finally, it should ultimately be considered that several federated clouds with real users and real usage are deployed. Such large-scale cloud deployments need to have an extended implementation of a communication middleware for the coordination in a network of super nodes, complemented by additional services, to fully achieve an incentive-based resource assignment. For such systems, additional work is needed to develop in detail the feedback loop between the user's contribution and

the experience the user obtains from the cloud services, needed for the building and maintenance of a cloud in community networks.

6. Related Work

The idea of collaboratively built community clouds follows on from earlier distributed voluntary computing platforms, like BOINC [21], Folding@home [22], PlanetLab [23] and Seattle [24], which mainly rely on altruistic contribution of resources from the users, though various mechanisms have been studied in the context of peer-to-peer systems [17] that address different problems of collaborative resource sharing. There are only a few research proposals for community cloud computing [3]. Most of them do not go beyond the level of an architecture, and at most a practical implementation is presented. None of these implementations, to our knowledge, are actually being deployed inside of real community networks.

The Cloud@Home[25] project aims to harvest resources from the community for meeting the peaks in demand, working with public, private and hybrid clouds to form cloud federations. The authors propose a rewards and credit system for ensuring quality of service. Social cloud computing [26] takes advantage of the trust relationships between members of social networks to motivate contribution towards a cloud storage service. Users trade their excess capacity to earn virtual currency and credits that they can utilize later, and consumers submit feedback about the providers after each transaction which is used to maintain reputation of each user. Social clouds have also been deployed in CometCloud framework by federating resources from multiple cloud providers [27]. Social compute cloud [28], implemented as an extension of Seattle [24] platform, enables the sharing of infrastructure resources between friends connected through social networks, and explores bidirectional preference-based resource allocation.

Among federated cloud infrastructures, Gall et al. [29] have explored how an InterCloud architecture [30] can be adapted to community clouds. Esposito et al. [31] present a flexible federated Cloud architecture based on a scalable publish and subscribe middleware for dynamic and transparent interconnection between different providers. Zhao et al. [32] explore efficient and fair resource sharing among the participants in community-based cloud systems. Jang et al. [33] implement personal clouds that federate local, nearby and remote cloud resources to enhance the services available on mobile devices. Palmieri et al. [34] use agent-based game-theoretic scheme for scheduling computing resources between providers in federated cloud infrastructures.

From the review of related work, we find that none of the above cases correspond to the concrete situation of community networks such as targeted by us. In the cloud system that we propose, we aim to take into account several of the important factors that characterize community networks, such as the scenarios we identified from the conditions of community networks, and the cloud architecture we considered is tailored to these scenarios. We also put emphasis on implementing the proposed components as prototype and test them in deployments on real nodes to identify practical issues. We note that compared with the related work, we follow the approach of developing a prototype that should contribute to building a production community cloud system to be used in real community networks.

7. Conclusion and Future Work

Community clouds are motivated by the additional value they would bring to community networks. Applications and services deployed upon community clouds would boost the usage and spread of the community network model as ICT infrastructure for society. As such, it is timely to research on clouds for community networks, since mainstream cloud computing technologies are mature now and are widely used in today's Internet.

The paper analyses the key socio-technical characteristics of community networks in order to derive two community cloud scenarios, the local community cloud and the federated community cloud. These scenarios are targeted by a community cloud architecture which is proposed, with the need for an incentive-driven regulation mechanism identified as a key component to encourage contribution towards and foster adoption of community clouds. The regulation component is implemented as a prototype, and evaluated in an experimental deployment on real community network nodes to explore its behaviour for both community cloud scenarios. This incentive mechanism is implemented in a simulator in order to be able to perform assessments for large scale scenarios. With simulation experiments we characterized the behaviour of different settings of the incentive mechanism, and evaluated the success ratio of nodes and resource utilization. A deeper analysis of the behaviour allowed us to better understand the influence of the different configuration options.

Carrying onwards from the experience and results with this prototype, a working service needs to be developed further that provides the feedback loop between the users' contribution and experience, which will be inevitable for adoption, sustainability, maintenance and growth of cloud infrastructures in community networks. Larger scale deployments are required with extended implementation of the different components of the community cloud architecture. This should be complemented by additional services and applications deployed in the cloud infrastructure, which will provide enhanced value and utility to the members of community networks for their contribution towards the community cloud.

References

[1] Guifi.net new sections of fiber deployed to the farm (2012).
URL http://en.wikinoticia.com/Technology/internet/122595

[2] P. Mell, T. Grance, The NIST Definition of Cloud Computing, NIST Spec. Publ. 800 (145).

[3] A. Marinos, G. Briscoe, Community Cloud Computing, in: Cloud Comput. First Int. Conf. CloudCom 2009, Vol. 5931 of LNCS, Springer Berlin Heidelberg, Beijing, China, 2009, pp. 472–484. arXiv:0907.2485.

[4] R. Rahman, et al., Improving Efficiency and Fairness in P2P Systems with Effort-Based Incentives, in: IEEE Int. Conf. Commun. (ICC 2010), IEEE, Cape Town, South Africa, 2010, pp. 1–5.

[5] D. Vega, R. Messeguer, S. F. Ochoa, F. Freitag, Sharing Hardware Resources in Heterogeneous Computer-Supported Collaboration Scenarios, Integr. Comput. Aided. Eng. 20 (1) (2013) 59–77.

[6] D. Vega, L. Cerda-Alabern, L. Navarro, R. Meseguer, Topology patterns of a community network: Guifi.net, in: 1st Int. Work. Community Netw. Bottom-up-Broadband (CNBuB 2012), within IEEE WiMob, IEEE, Barcelona, Spain, 2012, pp. 612–619.

[7] Guifi.net's Forum and Mailing Lists (2014).
URL http://guifi.net/en/forum

[8] Wireless Commons License for Open, Free & Neutral Network (OFNN) (2010).
URL http://guifi.net/es/ProcomunXOLN

[9] R. Moreno-Vozmediano, R. S. Montero, I. M. Llorente, IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures, Computer 45 (12) (2012) 65–72.

[10] A. M. Khan, F. Freitag, Exploring the Role of Macroeconomic Mechanisms in Voluntary Resource Provisioning in Community Network Clouds, in: Distrib. Comput. Artif. Intell. 11th Int. Conf., Vol. 290 of Advances in Intelligent Systems and Computing, Springer International Publishing, Salamanca, Spain, 2014, pp. 269–278.
[11] Pico Peering Agreement v1.0 (2005).
URL http://www.picopeer.net
[12] OpenStack: Open Source Cloud Computing Software (2014).
URL http://www.openstack.org/
[13] A. M. Khan, U. C. Buyuksahin, F. Freitag, Prototyping Incentive-based Resource Assignment for Clouds in Community Networks, in: 28th IEEE Int. Conf. Adv. Inf. Netw. Appl., IEEE, Victoria, Canada, 2014.
[14] A. M. Khan, U. C. Buyuksahin, F. Freitag, Towards Incentive-based Resource Assignment and Regulation in Clouds for Community Networks, in: Econ. Grids, Clouds, Syst. Serv., Vol. 8193 of LNCS, Springer International Publishing, Zaragoza, Spain, 2013, pp. 197–211.
[15] A. M. Khan, M. Selimi, F. Freitag, Towards Distributed Architecture for Collaborative Cloud Services in Community Networks, in: 6th Int. Conf. Intell. Netw. Collab. Syst., IEEE, Salerno, Italy, 2014.
[16] M. Bina, G. Giaglis, Unwired Collective Action: Motivations of Wireless Community Participants, in: Int. Conf. Mob. Bus., IEEE, Copenhagen, Denmark, 2006, pp. 31–31.
[17] X. Shen, H. Yu, J. Buford, M. Akon, Handbook of Peer-to-Peer Networking, Vol. 1, Springer Heidelberg, 2010.
[18] J. C. Anderson, J. Lehnardt, N. Slater, CouchDB: The Definitive Guide, 1st Edition, O'Reilly Media, Inc., 2010.
[19] OpenWrt: Linux distribution for embedded devices (2014).
URL http://openwrt.org/
[20] B. Braem, R. Baig Viñas, A. L. Kaplan, A. Neumann, I. Vilata i Balaguer, et al., A case for research with and on community networks, ACM SIGCOMM Comput. Commun. Rev. 43 (3) (2013) 68–73.
[21] D. P. Anderson, BOINC : A System for Public-Resource Computing and Storage, in: 5th IEEE/ACM Int. Work. Grid Comput., Pittsburgh, USA, 2004, pp. 4–10.
[22] A. L. Beberg, et al., Folding@home: Lessons From Eight Years of Volunteer Distributed Computing, in: 8th IEEE Int. Work. High Perform. Comput. Biol. (HiCOMB '09), within IPDPS, IEEE, Rome, Italy, 2009.
[23] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, M. Bowman, PlanetLab: An Overlay Testbed for Broad-Coverage Services, ACM SIGCOMM Comput. Commun. Rev. 33 (3) (2003) 3–12.
[24] J. Cappos, I. Beschastnikh, A. Krishnamurthy, T. Anderson, Seattle: a platform for educational cloud computing, in: 40th ACM Tech. Symp. Comput. Sci. Educ. (SIGCSE '09), ACM, Chattanooga, USA, 2009, pp. 111–115.
[25] S. Distefano, A. Puliafito, Cloud@Home: Toward a Volunteer Cloud, IT Prof. 14 (1) (2012) 27–31.
[26] K. Chard, K. Bubendorfer, S. Caton, O. F. Rana, Social Cloud Computing: A Vision for Socially Motivated Resource Sharing, IEEE Trans. Serv. Comput. 5 (4) (2012) 551–563.
[27] M. Punceva, I. Rodero, M. Parashar, O. F. Rana, I. Petri, Incentivising resource sharing in social clouds, Concurr. Comput. Pract. Exp.
[28] S. Caton, C. Haas, K. Chard, K. Bubendorfer, O. F. Rana, A Social Compute Cloud: Allocating and Sharing Infrastructure Resources via Social Networks, IEEE Trans. Serv. Comput. 7 (3) (2014) 359–372.
[29] M. Gall, A. Schneider, N. Fallenbeck, An Architecture for Community Clouds Using Concepts of the Intercloud, in: 27th Int. Conf. Adv. Inf. Netw. Appl. (AINA '13), IEEE, Barcelona, Spain, 2013, pp. 74–81.
[30] R. Buyya, R. Ranjan, R. N. Calheiros, InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services, Algorithms Archit. Parallel Process. 6081 (2010) 20–31.
[31] C. Esposito, M. Ficco, F. Palmieri, A. Castiglione, Interconnecting Federated Clouds by Using Publish-Subscribe Service, Cluster Comput. 16 (4) (2013) 887–903.
[32] H. Zhao, X. Liu, X. Li, Towards efficient and fair resource trading in community-based cloud computing, J. Parallel Distrib. Comput. 74 (11) (2014) 3087–3097.
[33] M. Jang, K. Schwan, K. Bhardwaj, A. Gavrilovska, A. Avasthi, Personal clouds: Sharing and integrating networked resources to enhance end user experiences, in: 33rd Annu. IEEE Int. Conf. Comput. Commun., IEEE, Toronto, Canada, 2014, pp. 2220–2228.
[34] F. Palmieri, L. Buonanno, et al., A distributed scheduling framework based on selfish autonomous agents for federated cloud environments, Futur. Gener. Comput. Syst. 29 (6) (2013) 1461–1472.